Risk Management the Key to Successful Reuse

Elizabeth L. Burd, John A. McDermid

Department of Computer Science,
University of York,
Heslington,
York,
YO1 5DD,
UK

Tel: (+44) 904 432711
Fax: (+44) 904 432767
Email: liz@minster.york.ac.uk

Abstract

In this position paper we stress the importance of risk management to the process of software reuse. We consider the problems that are associated with reuse and have accounted for its failure to become an important and respected part of the software development process. We discuss, in brief, the work that we have carried out. In particular we concentrate our efforts for this paper on the procedure of reusing risk assessments to direct a new software development. We then look at some of the implications that increasing a developer's knowledge of risks has on the acceptance of reuse to project managers. Finally we indicate the new directions our work will take.

Keywords: Reuse, Risk, Software life cycle, Process models.

Workshop Goals: Listening , learning, discussing and meeting people

Working Groups: Reuse process models, Reuse management, organisation and economics, Reuse maturity models, Education.

1 Background

From a study of the research conducted into the problems associated with software reuse and also from the industrial experiences of companies trying to employ reuse we, in harmony with others, have identified a basis on which the success of reuse depends. We believe that for reuse to flourish, it must be viewed as an integral part of the software life cycle. Changes must be made to the way in which industry views reuse. Industry needs to move away from simply the reuse of code and certainly away from the attitude that it is acceptable not to consider reuse at all. We examined why these attitudes have existed and our studies have led us to the conclusion that there are a number of inhibiting factors which limit the appeal of reuse to project managers and software developers. These inhibiting factors include, at a technical level:

- *development* - Knowing what kind of software is reusable and as equally difficult is knowing how to develop a software component which is potentially reusable.
- *storage* - Once we have developed an item of reusable software how, and where, should it be retained for future retrieval and reuse.
- *retrieval* - If we are to reuse software then we must be able to easily find what we require matching what is available with our needs.
- *verification* - How can we be sure that the component which we are proposing to reuse actually performs the functions that it claims it does in the environment in which we use it.
- *evaluation* - How can we judge that the functions we require from our reusable unit and the functions that it provides are the same.
- *modification* - If our evaluations have shown that differences exist in the reuse unit's and our requirements then how do we perform the necessary modifications and what effect will this have on the reusable unit including the results of previous verifications.
- *integration* - What will the effect be on the reusable unit of attempting to integrate it into our development.

We thus came to the realisation that the underlying technical reason why reuse was not as successful is that developers would have liked as the inhibiting factors which we have described above increase development risks. Since project managers tend towards the path with the smallest risks they thus often, and rightly given the circumstances, tend to develop 'from scratch'. We therefore consider that if reuse is to be successful we need to indicate its tangible benefits to project managers by reducing the risks associated with development and especially when reuse is being considered. In addition it should strive to make reuse a continuous and integral part of the software development process as this has an effect both on risks and acceptability of risks.

We acknowledge that there are other factors affecting the take up of reuse e.g. individual motivations but such issues are outside the realm of this paper.

2 Comparison

Barnes and Bollinger [1, 2] have indicated similar beliefs about the nature of the problems that are associated with software developments incorporating reuse. They advocate the necessity for a broad spectrum of reuse to be considered and stress the importance of considering human problem solving as an aspect of software development that is potentially reusable. Their work has concentrated on the need for incentives for performing reuse and have indicated ways in which this can be taken account of in the software development process. In addition they have indicated that life cycle integration is likely to one of the key factors which makes reuse truly effective. This work formed the foundation from which we embarked in our research One major contribution, and the issue on which we focus here, is the notion of risk and the reuse of risk assessment - a sort of human problem solving.

In addition we have also built our work around Boehm's spiral model [3, 4]. We will consider the additions we have made to the model in more detail below, but in brief we have to base our work on the spiral model because of its risk based approach to software development. Our extensions were included to enhance the work of Boehm's to encourage a life cycle approach to reuse and a broad definition of reuse.

3 Position

We have indicated above that development risks are actually be increased with reuse. This is probably the most compelling reason to develop systems 'from scratch'. However, the perceptions of risk may outweigh the actual risks, and social factors will tend to lead project managers towards the (perhaps illusory) feeling that developing 'from scratch' puts then in control' and hence minimises risk. We consider these apparent increased risks an unnecessary burden for reuse to carry and believe that given a broader definition of what is reusable from the software development process then reuse can actually serve to reduce development risks. Perhaps more importantly by enabling risk-based decision making about reuse to be explicit, we can help ensure that decisions about reuse are balanced and well supported.

3.1 Hypothesis

The hypothesis of the PROM project was therefore: reusing all products of software development will reduce the risks that are associated with reuse and that since developers will find reuse more acceptable, if they are aware of the scale of risks that are involved then reuse will become a necessary and informative part of software development.

3.2 Research

As we indicated above that our work has been based on building extensions to Boehm's spiral model. This allows us to incorporate our aims of providing a reuse method which is integral to software development while supporting reuse continually and also encouraging a risk based approach to development. These extensions have taken the form of a 'reuse spiral'. This has a symbiotic relationship the spiral model; the spiral model encourages a risk based approach to development and the reuse spiral provides the 'know-how' for reuse. Thus the information which is collected by one of the spirals is used by the other and vice versa.

Each of the spirals of the two models are split into four quadrants. The activities that are concerned with these quadrants are as follows:

Spiral model	Reuse spiral	Problems tackled
Q1 Define system objectives and constraints	Use information gathered as search criteria	Retrieval
Q2 Evaluate alternatives identify and resolve risks.	Reuse risk assessment information to aid risk management	Evaluation Verification
Q3 Develop and verify next - level product	Reuse products of previous developments where possible	Development Modification Integration
Q4 Plan next phases	Focus record and commit decisions to the library	Storage

3.3 Case Study

We consider that one area which is often disregarded in the field of reuse research are those activities which are concerned with the second quadrant of the reuse spiral; mainly that of the evaluation of the risks associated with a development. We have indicated how important this is for successful software development and therefore we will illustrate this process, of reusing risk evaluations, with the aid of an example derived from a case study of the National Health Service.

In the second quadrant we perform the following activities:

1. We use information from the similar projects which were located in the first quadrant and investigate what options are available for developing a system, and secondly what risk assessments activities were associated which each option. From this information we are able to indicate some of the risk criteria which are associated with our new development. In terms of the health service, typical risk criteria for developing a system which informs doctors which drugs are functionally equivalent may be: the doctors reaction to the system; the cost of development; the accuracy of the information it generates; or the feasibility of building the system. Typical options may be to: develop a database application, an expert system or perhaps nothing at all i.e. retain a manual system.
2. We can now use sensitivity analysis to allow us to evaluate which is the best option to take for our proposed development. In order to do this then we evaluate risk criteria individually to obtain the utility for each risk criteria. The risk criteria for each option may be the same but their utility will be different. (Here we use utility in a technical sense, a concept unifying benefit and risk.) In the case of our drug substitution program the utility for cost will depend on the expected cost of the individual options and the utility for doctors acceptance of the system may for instance be greater for the database option than the expert system since doctors may feel an expert system is de-skilling their job. Weightings are then placed on

the individual criteria (again information gained from previous projects) in order that the interaction effects between the individual risk criteria can be evaluated.

3. By folding back the decision trees the utility for each option can then be considered and the option with the highest utility, or greatest expected payoff, can be selected.

3.4 Conclusions

Our studies have lead us to conclude that an all round approach to reuse improves the chances of a product or process definition of a previous development being reusable. Without taking a risk based approach to development the type of reuse which we have described would in most cases be infeasible. Thus we believe that reusing risk assessment, weightings and prototypes etc. allow more informed judgements to be made as to the feasibility of various options of software development and therefore reuse to be, in general, more successful.

References

[1] T. B. B.H. Barnes, "Making Reuse Cost-Effective," *IEEE Software*, pp. 13–24, January 1991.

[2] S. P. T.B. Bollinger, "Economics of Reuse: issues and alternatives," *Information and Technology*, December 1990.

[3] B. Boehm, "A Spiral Model for Software Development and Enhancements," *IEEE Computer*, pp. 61–72, May 1988.

[4] B. Boehm, "Software Risk Management: Principles and Practices," *IEEE Software*, pp. 32–41, January 1991.

4 Biography

Elizabeth Burd's background is in education; she is a qualified teacher. In 1989 she entered the research community and after completing a masters in Information processing she subsequently proceeded to research for a Ph.D. in the Information Systems group at the University of York. She is currently working on the PROM project which is concerned with a life cycle approach to reuse. Her interests in education are very much reflected in the options she is considering to aid some of the problems associated with reuse.

Elizabeth is also a member of the British Computer Society Reuse Special Interest Group committee.

John McDermid has been Professor of Software Engineering at the University of York since 1987. Prior to coming to York he worked for SD-Scicon and the Royal Signals and Radar Establishment (now the DRA Malvern). At the University he runs the High Integrity Systems Engineering group, concentrating on safety-critical and dependable systems. He is also the technical director of the Dependable Computing Systems Centre set up by British Aerospace at the Universities of York and Newcastle and a director of York Software Engineering Ltd.

He is active in the professional community, and has assisted the IEE and BCS in major studies of safety-critical systems. He has published widely, being the author or editor of six books, and the author or co-author of about one hundred and twenty papers and articles.